IEEE Std 1609.2-2016 Guidance Note 3: Replay detection

William Whyte, Security Innovation

Version 0.1, 2016-07-20

1 Introduction

This note identifies an area in IEEE Std 1609.2-2016 that has been reported as a security defect that does not affect interoperability, and gives guidance to implementers and certification testers to avoid the defect.

The defect concerns replay protection: the mechanisms specified for replay protection in fact allow a small number of replay attacks to be carried out. The guidance defeats this attack. The defect and its correction do not affect interoperability.

This guidance note is structured as follows. Section 2 provides a summary of the issue, the impact, and current practice. Section 3 provides the recommended implementation change. Section **Error! Reference source not found.** recommends a corrigendum to 1609.2 to fix this defect and provides potential replacement text that should be considered for. Section 5 provides acknowledgements.

2 Summary of issue

2.1 How ECDSA signatures plus 1609.2's replay detection allow a replay attack

1609.2 clause 5.2.4.2.6, "Replay", says in its entirety:

> The SDS can be requested to detect whether a signed SPDU is an exact duplicate of one previously processed by the SDS for that SDEE. The replay detection service is provided by SSME-Sec-ReplayDetection.request, SSME-Sec-ReplayDetection.confirm. The replay detection service indicates that a signed SPDU is a replay if the entire encoded signed SPDU, including signature and other fields such as generation time inserted by the SDS, is identical to a recently received SPDU. The definition of "recently received" is SDEE-specific, but it is a logically consistent choice for this value to be the same as the value used to determine whether a SPDU has a generation time too far in the past (see 5.2.4.2.2), and the interfaces defined in this standard enforce that the two values are the same. Other replay detection techniques, such as ones based on the payload only or on the same data encoded in different ways, are out of scope of this standard.
>
> Whether or not the SDS carries out replay detection, and the length of the interval for replay detection, is part of the SDEE specification. The 1609.2 security profile (see Annex C) is provided to specify replay detection behavior by the security services. Annex C also provides discussion of how to establish whether replay detection is important for a particular SDEE.

The key statement here is that the signed SPDU is a replay if "the entire encoded signed SPDU, **including signature** …, is identical to a recently received SPDU". This is a problem because of an unexpected property of ECDSA signatures. A valid ECDSA signature is a pair of integers (r, s) that

are cryptographically linked to the message m (or more specifically, its hash H(m)) and the public key A. It turns out that **if (r, s) is a valid signature for A on m, then (r, -s) is also a valid signature** for the same message and public key. This means that if an attacker sees a validly signed message, they can create another version of the message that is validly signed but has a different signature. Since replay detection is carried out on the whole message, including the signature, the new message will not be caught by the replay detection and **the attacker can successfully replay each message once**.

2.2 How the SignerIdentifier structure plus 1609.2's replay detection allow a replay attack

The SignerIdentifier structure lets receiver of a signed message know which certificate to use to verify the message. It has the following form in ASN.1:

```
SignerIdentifier ::= CHOICE    {
  digest           HashedId8,
  certificate      SequenceOfCertificate,
  self             NULL,
  ...
}
```

It is included in every signed SPDU:

```
SignedData ::= SEQUENCE  {
  hashId           HashAlgorithm,
  tbsData          ToBeSignedData,
  signer           SignerIdentifier,
  signature        Signature
}
```

The signature is calculated over a hash derived from:

- the COER encoding of the tbsData field canonicalized according to the encoding considerations given in IEEE 1609.2 clause 6.3.6.
- the COER-encoding of the Certificate that is to be used to verify the SPDU, canonicalized according to the encoding considerations given in IEEE 1609.2 clause 6.4.3.

Note that while the signing certificate is included in the signature calculation, the exact transmitted representation of the SignerIdentifier is not.

For a given (message, signature) pair, the fact that the SignerIdentifier is a choice means the signing certificate can be represented in multiple ways:

- With the digest form of SignerIdentifier, in which case it is sent as an eight-byte hash of the certificate
- As the certificate itself

- As a certificate chain of length 2, containing the signing certificate and the issuing CA's certificate
- As a certificate chain of length 3…

If an attacker takes an Ieee1609Dot2Data containing the digest form of the SignerIdentifier and replaces it with one containing the pseudonym certificate:

- The replay detection algorithm will not trigger (recall that it only triggers if "the entire encoded signed SPDU… is identical to a recently received SPDU"; changing the contents of the SignerIdentifier will mean that the encoded SPDU is not identical to the previously received one).
- The signature will still verify (because the encoding of the SignerIdentifier is not input to the signature calculation)

This means that if an attacker knows the signing certificate and (optionally) any of the certificates immediately above it in the chain, **the attacker can successfully replay each message once for each known certificate at the bottom of the chain** up to the limit of the chain length permitted by the security profile.

2.3 Impact

A replay attack is an attack where an attacker is able to get a victim to act twice on the same message – for example, to pay the attacker twice. In a context where acting twice on the same message is harmful, replay attacks can be very serious. **1609.2 should be corrected so this attack is no longer possible.**

However, in a 1609.2 setting, not all repeated messages are replays. For example, WSAs may be exactly repeated, including their signatures, without being considered a replay. The 1609.2 security profile allows each application specifier to say whether or not replay is to be detected by the security services in the context of that application. It turns out that in the BSM security profile contained in J2945/1, replay is not checked by the security services; instead, replayed messages are just another type of bad data that the application logic is expected to handle. Therefore, **this attack does not affect BSM handling**. The attack needs to be addressed to protect other applications, but existing BSM implementations should not be vulnerable.

2.4 Recommended mitigation with rationale

Currently, conformant 1609.2 implementations carry out replay detection based on the received Ieee1609Dot2Data. This is vulnerable to replay attacks that modify the signer identifier, and to replay attacks that modify the signature.

Possible mitigations are:

- Modify the representation of the Ieee1609Dot2Data used for replay detection by
    - omitting OR canonicalizing the signer identifier;
    - omitting OR canonicalizing the signature

- Modify the input to the signature itself by including the transmitted SignerIdentifier in the hash (this addresses the malleable SignerIdentifier issue but not the malleable signature issue)

("Canonicalize" means "put into a defined encoding such that only one encoding is correct").

We do not consider modifying the input to the signature as a realistic option because (a) this would cause backwards incompatibility with existing 1609.2 implementations and (b) this wouldn't address the signature malleability issue. This leaves only the option of modifying the representation of the Ieee1609Dot2Data used for replay detection.

Omit SI/omit signature: If the signer identifier and signature are omitted from replay detection, that means that if two messages with the same payload are sent by different senders, one will be rejected. This is not desired behavior.

Omit SI/canonicalize signature: Signatures created by different signers will be different, so this essentially eliminates the risk that that if two messages with the same payload are sent by different senders, one will be rejected. This option is therefore potentially acceptable, although the canonicalization for the signature needs to be defined.

Canonicalize SI/omit signature: This ensures that the same valid messages from different senders will not have one message rejected. Note also that when an SPDU is hashed for signing or verification, the actual octet string input to the asymmetric operation is not a direct hash of the message but instead

Hash (Hash (*data*) || Hash (*signing certificate*))

This is exactly the information needed to detect replays while allowing valid repeats, so since it's being calculated anyway, it's very attractive to use it for replay detection as well.

Canonicalize SI/canonicalize signature: This also ensures that the same valid messages from different senders will not have one message rejected. This option is therefore potentially acceptable, although the canonicalization for the signature needs to be defined.

Conclusion. None of the approaches that involve canonicalizing the signature have compelling advantages over omitting the signature from the hash. **We therefore recommend that the octet string used for replay detection is the SignedData, with a canonicalized SignerIdentifier, and with the signature omitted.** This can be stored as the same octet string input to the asymmetric operation when signing and verifying, i.e. Hash (Hash (*data*) || Hash (*signing certificate*))

3 Recommended implementation

The recommended implementation of 1609.2 is that replay detection detects SPDUs in which both of the following are repeated:

- the COER encoding of the tbsData field canonicalized according to the encoding considerations given in IEEE 1609.2 clause 6.3.6.
- the COER-encoding of the Certificate that is to be used to verify the SPDU, canonicalized according to the encoding considerations given in IEEE 1609.2 clause 6.4.3.

4 Recommended change to the standard (preliminary)

We recommend that a corrigendum is issued to the standard to close off this security hole. One possible wording is to replace the opening sentences of 5.2.4.2.6 with

> The SDS can be requested to detect whether a signed SPDU is a duplicate of one previously processed by the SDS for that SDEE. The replay detection service is provided by SSME-Sec-ReplayDetection.request, SSME-Sec-ReplayDetection.confirm. The replay detection service indicates that a signed SPDU is a replay if BOTH the COER encoding of the tbsData field canonicalized according to the encoding considerations given in IEEE 1609.2 clause 6.3.6, AND the COER-encoding of the Certificate that is to be used to verify the SPDU, canonicalized according to the encoding considerations given in IEEE 1609.2 clause 6.4.3, are identical to those information elements for another recently received SPDU.

Additionally, changes need to be made to clause 9.5.1, which describes the replay protection primitives defined on the SSME-SEC SAP.

Clause 9.5.1.1.1 should be revised to replace “duplicate of signed data” with “replay of signed data”.

Clause 9.5.1.1.2 should alter the description of the input value *Data* from “The encoded SignedData that is to be checked for being a replay” to “The encoded ToBeSignedData and signing certificate that are to be checked for being a replay”.

Other changes may also be necessary following a more complete review of the standard.

5 Acknowledgements

Thanks to the Car2Car Communications Consortium Security Working Group for the observation that if (r, s) is a valid signature then so is (r, -s), and thanks to Mike Cox of Security Innovation who was first to observe that this enabled the signature-malleability-based replay attack.